Using Reproducing Kernel Hilbert Spaces to Improve Support Vector Machines

Abhishek Devarajan

December 2, 2021

Abstract

The sharp rise in the availability of data and the demand for said data to be analyzed, has lead to a myriad of different machine learning algorithms. One such algorithm, praised for being simplistic yet highly effective, is the Support Vector Machine (SVM). The secret behind SVMs success is the so called "kernel trick"– a clever use of the Riesz Representation Theorem and Hilbert Spaces that turns complicated data into linearly separable sets. In this paper, we will cover the properties of general Hilbert Spaces and Reproducing Kernel Hilbert Spaces specifically. We will then describe the Riesz Representation theorem and kernel functions. Putting all of these parts together, we conclude with an outline of the kernel trick and demonstrations of its effectiveness on toy datasets.

1 Introduction

As machine learning becomes more prevalent in multiple industries, there is a big push to build large neural networks to perform classification and regression tasks. In some sense, this gravitation towards neural networks is natural, as these networks are capable of approximating any given function, provided that enough layers are added [5]. However, one should be wary about using neural networks when other, simpler algorithms are available. In part, this is due to the fact that neural networks are very computationally expensive. Additionally, the increase in neural network usage has rapidly outpaced the theory that describes the properties and the convergence of these networks. For these reasons, it is worthwhile to employ less complex algorithms that are better understood from a theoretical standpoint.

In particular, we will use this paper to discuss the Support Vector Machine (SVM) algorithm for binary classification problems. SVM is an algorithm that is not only simplistic in practice, but also very intuitive in concept. Essentially, SVM attempts to classify points in an n dimensional space by drawing a hyperplane through the dataset. Any points on one side of the hyperplane correspond to one class, while all of the points on the other side of the hyperplane are assigned to the other class. From this description, it should be clear that SVM performs best when the points in a dataset are disjoint, convex sets [3]. In cases where the data points are not separable by a hyperplane (i.e. linearly separable), SVM will always fail to classify at least one point correctly. If this is the case, how can SVM compete with more powerful algorithms such as neural networks? The answer to this question lies with the so called "kernel trick".

The kernel trick is an application of functional analysis that allows SVM to accurately classify non-linearly separable data. Interestingly, the underlying theory behind the kernel trick dates back to the 1950s, well before the advent of machine learning. The underlying principle behind the kernel trick is that there must exist certain maps, known as "feature maps", that allow us to transform our data points into a higher-dimensional Hilbert Space (feature space) in which they are linearly separable. Although this seems like a simple premise, it is often difficult to define these feature maps explicitly. Furthermore, storing large, potentially infinite, vectors in computer memory is often very intensive. To circumvent these issues, we introduce special kernel functions and show that constructing a kernel is equivalent to defining a feature map and its corresponding feature space.

The rest of the paper will proceed as follows. In Section 2, we provide definitions and explanations that are prerequisite to understanding the kernel trick. In Section 3, we state and prove key theorems that facilitate the kernel trick. Section 4 covers the process of the kernel trick. Finally, Section 5 will provide examples of kernel functions and demonstrate the effect of applying the kernel trick to non-linearly separable data.

2 Background

In this section, we will cover the prerequisite knowledge needed to understand the kernel trick and its application for SVM. We start with a description of SVM and its loss function and then continue by discussing a number of definitions leading up to that of a RKHS.

Definition 2.1 (Support Vector Machine)**.** Consider a dataset of n points, $D = \left\{ \left(x_{(i)}, y_{(i)} \right) \mid i = 1, \ldots, n \right\}$ where each $x_{(i)}$ is a point in an arbitrary, non-empty input space $\mathcal{X}$ and each label $y_{(i)}$ is an element of $\mathcal{Y} = \{-1, 1\}$. Typically, D is referred to as the training set. Define the hyperplane $h(x)$ as

$$h(x) = \langle w, x \rangle + b$$

for some $w \in \mathcal{X}$ and $b \in \mathbb{R}$. The **Support Vector Machine (SVM)** algorithm involves choosing an optimal w and b such that the sign of $h\left(x_{(i)}\right)$ is equal to the sign of $y_{(i)}$. Now say we have a point $z \in \mathcal{X}$ where $z \neq x_{(i)}$ for any $i = 1, \ldots, n$. To classify such a point, we take the sign of $h(z)$ and assign z the label in $\mathcal{Y}$ with the same sign.

Remark 2.2. Throughout this paper we use the subscript (i) to denote points within a dataset. The subscript i is reserved for the i-th element of a vector or sequence.

Definition 2.3 (SVM Loss Function)**.** Since complete linear separation is not always possible, we loosen the restriction of having every data point accurately classified. We formulate the **loss function** of SVM as the primal optimization problem

$$\begin{aligned} \min\ & \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.}\ & y_i \left(\left\langle w, x_{(i)} \right\rangle + b \right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \end{aligned}$$

where C is a fixed constant and each ξ_i is a slack variable. The slack variables allow us to mislabel a few points for the sake of higher overall accuracy. In essence, this optimization problem attempts to create a hyperplane that classifies the training data as accurately as possible, while maximizing the distance between the hyperplane and the data points (i.e. the "margin"). The reason for this is to ensure that small perturbations in the training data do not drastically affect the accuracy of the classifier. For the kernel trick, we formulate the corresponding **dual problem**,

$$\begin{aligned} \max\ & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_{(i)} y_{(j)} \alpha_i \alpha_j \left\langle x_{(i)}, x_{(j)} \right\rangle \\ \text{s.t.}\ & \sum_{i=1}^{n} \alpha_i y_{(i)} = 0. \end{aligned} \tag{1}$$

For now, it suffices to say that this dual problem is equivalent to its primal counterpart [3, 9]. In Section 4 we will revisit Equation (1) and see its relationship with the kernel trick in more detail.

Definition 2.4 (Hilbert Space)**.** A **Hilbert Space** $\mathcal{H}$ is a real or complex inner product space such that $\mathcal{H}$ is also a complete metric space with respect to the distance function induced by its inner product.

Remark 2.5. Completeness is a property whose importance is not always obvious when constructing theorems around Hilbert Spaces. Being complete implies that every Hilbert Space has an orthonormal basis [6], which is a property often taken for granted when working in finite dimensional spaces. Crucially for this paper, completeness ensures that we can decompose a Hilbert Space into a direct sum between a closed subspace and its orthogonal complement [6].

Example 2.6 (Types of Hilbert Spaces)**.** The following are common examples of Hilbert Spaces used in various applications.

- Every finite dimensional inner product space with field $\mathbb{R}$
- $\ell_2 = \{(a_1, a_2, \dots) \mid a_i \in \mathbb{C}, \sum_{i=1}^{\infty} |a_i|^2 < \infty\}$ with $\langle a, b\rangle = \sum_{i=1}^{\infty} \overline{b_i} a_i$
- $L^2(\mathbb{R}) = \left\{ f : \mathbb{R} \to \mathbb{R} \;\middle|\; \int_{-\infty}^{\infty} |f(x)|^2 dx < \infty \right\}$ with $\langle f, g\rangle = \int_{-\infty}^{\infty} g(x)f(x)dx$

Hilbert Spaces provide us with a set of very nice and intuitive properties thanks to their completeness and inner products. For our purposes, we will build on the structure of a general Hilbert Space and construct an RKHS. From this point on we will limit our discussion to real Hilbert Spaces for simplicity, although all of the topics discussed can be generalized to work with complex Hilbert Spaces as well.

Definition 2.7 (Linear Evaluation Functional)**.** Let $\mathcal{H}$ be a Hilbert Space of functions $f : \mathcal{X} \to \mathbb{R}$ and consider the linear functional $\mathcal{L}_x : \mathcal{H} \to \mathbb{R}$ given by

$$\mathcal{L}_x(f) = f(x).$$

$\mathcal{L}_x$ is known as the **Linear Evaluation Functional (LEF)**.

Definition 2.8 (Reproducing Kernel Hilbert Space)**.** Suppose $\mathcal{H}$ is a Hilbert Space of real-valued functions with domain $\mathcal{X}$, where $\mathcal{X}$ is any arbitrary set. We consider a linear functional, $\varphi : \mathcal{H} \to \mathbb{R}$ to be **bounded** if there exists a real number M such that

$$|\varphi(f)| \leq M||f||.$$

If the linear evaluation functional $\mathcal{L}_x$ on $\mathcal{H}$ is bounded for all $x \in \mathcal{X}$, then $\mathcal{H}$ is a **Reproducing Kernel Hilbert Space (RKHS)**.

3 Representation Theorem

In this section, we will cover a few important theorems from functional analysis. These theorems provide the crucial link between the definitions established in Section 2 and the kernel trick.

Theorem 3.1 (Continuity of Bounded Linear Functionals)**.** *Let $\mathcal{H}$ and $\mathcal{L}_x$ be defined as they were in Definition 2.7. $\mathcal{L}_x$ is bounded if and only if $\mathcal{L}_x$ is continuous.*

The proof of this theorem is covered in any introductory functional analysis class and, as such, will not be covered in detail here. For a full proof, please refer to [8]. Now we will prove the Riesz Representation Theorem, which is used to write linear functionals as inner products.

Theorem 3.2 (Riesz Representation Theorem)**.** *Let $\mathcal{H}$ be a Hilbert Space with inner product $\langle \cdot, \cdot \rangle$. For every continuous functional $\varphi : \mathcal{H} \to \mathbb{R}$, there exists a unique $f_\varphi \in \mathcal{H}$ such that $\varphi(g) = \langle g, f_\varphi \rangle$ for all $g \in \mathcal{H}$.*

Proof. Let φ be a continuous linear functional with domain $\mathcal{H}$. We will first prove the existence of a f_φ that satisfies Theorem 3.2. First, note that if φ is identically zero– i.e. $\varphi(g) = 0$ for all $g \in \mathcal{H}$– then $\varphi(g) = \langle g, 0\rangle$ for all $g \in \mathcal{H}$, and we are done. Consider the case where φ is not identically zero. Let $\mathcal{N} = \{h \mid h \in \mathcal{H}, \varphi(h) = 0\} = \varphi^{-1}(\{0\})$ be the null space of φ. Since φ is continuous and $\{0\}$ is a closed subset of $\mathbb{R}$, the Closed Graph Theorem [7] states that $\mathcal{N}$ must also be closed. Consequently, $\mathcal{N}^{\perp}$– i.e. the orthogonal complement of $\mathcal{N}$– must contain at least one non-zero vector p [2]. Now define a vector u as follows:

$$\begin{aligned} & u = (\varphi(g))p - (\varphi(p))g \\ \implies & \varphi(u) = \varphi(g)\varphi(p) - \varphi(p)\varphi(p) = 0 \\ \implies & u \in \mathcal{N}. \end{aligned}$$

By orthogonality,

$$\begin{aligned}
0 &= \langle u, p \rangle \\
&= \langle (\varphi(g))p - (\varphi(p))g, p \rangle \\
&= \varphi(g) \langle p, p \rangle - \varphi(p) \langle g, p \rangle \\
\implies \varphi(g) &= \left\langle g, \frac{p\varphi(p)}{||p||^2} \right\rangle.
\end{aligned}$$

Setting $f_\varphi = \dfrac{p\varphi(p)}{||p||^2}$ completes the existence proof.

Now, we will show that f_φ is the unique element in $\mathcal{H}$ that satisfies Theorem 3.2. Suppose to the contrary that there exists two functions f_1, f_2 such that $\phi(g) = \langle g, f_1 \rangle = \langle g, f_2 \rangle$, for all $g \in \mathcal{H}$.

$$\begin{aligned}
0 &= \varphi(g) - \varphi(g) \\
&= \langle g, f_1 \rangle - \langle g, f_2 \rangle \\
&= \langle g, f_1 - f_2 \rangle .
\end{aligned}$$

Since g is an arbitrary element in $\mathcal{H}$, take $g = f_1 - f_2$. This yields

$$\begin{aligned}
0 &= \langle f_1 - f_2, f_1 - f_2 \rangle \\
&= ||f_1 - f_2||^2 \\
\implies 0 &= f_1 - f_2.
\end{aligned}$$

□

Combining the definition of a RKHS with Theorem 3.1 and Theorem 3.2 indicates that we can write any evaluation functional as an inner product between the function being evaluated and a unique function in the RKHS. This is the so-called "reproducing property" that we use to carry out the kernel trick. We formalize this notion using the following notation.

Let $\mathcal{H}$ be a RKHS of functions $g : \mathcal{X} \to \mathbb{R}$ and let $\mathcal{X}$ be a non-empty set. For all $x \in \mathcal{X}$ and $f \in \mathcal{H}$, there exists a unique $K_x \in \mathcal{H}$ such that we can express the evaluation functional $\mathcal{L}_x$ as

$$\mathcal{L}_x(f) = \langle f, K_x \rangle .$$

4 Kernel Trick

In this section, we begin by presenting the problem that the kernel trick attempts to solve. We then move into the details and theoretical underpinnings of the kernel trick.

Problem 4.1. Suppose we have a training dataset $D = \{(x_{(i)}, y_{(i)})\}$ with each $x_{(i)} \in \mathcal{X}$ and each $y_{(i)} \in \mathcal{Y}$. Also suppose that D is non-linearly separable.

a) Using the information from earlier sections, does there exist a "feature map" ϕ that maps points in $\mathcal{X}$ to a new space (known as a "feature space") in which $D' = \{(\phi(x_{(i)}), y_{(i)})\}$ is linearly separable?

b) If such a ϕ exists, can we apply SVM to D' in the feature space? In particular, is

$$\begin{aligned}
\max \; & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_{(i)} y_{(j)} \alpha_i \alpha_j \left\langle x_{(i)}, x_{(j)} \right\rangle \\
\text{s.t.} \; & \sum_{i=1}^{n} \alpha_i y_{(i)} = 0
\end{aligned} \tag{2}$$

solvable?

To solve this problem, we must place a few restrictions on the feature map, ϕ. First, ϕ must be injective in order to prevent multiple data points from being mapped to the same point within the feature space. Another restriction is that ϕ must map the data points to a real inner product space in order to ensure that Equation (2) is solvable.

We begin by attempting to solve Problem 4.1 part a). Let $\mathcal{X}, \mathcal{H}$ be defined as before. Remember from Theorem 3.2 that $f(x) = \langle f, K_x \rangle$, where K_x is a unique element of $\mathcal{H}$. In particular, for some $z \neq x$, we have $f(z) = \langle f, K_z \rangle$, where $K_z \neq K_x$. Thus, there exists an injective mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $\phi(x) = K_x$, $\forall x \in \mathcal{X}$. Furthermore, since the feature space in this case is $\mathcal{H}$, we are able to calculate $\langle \phi(x), \phi(z) \rangle$.

Now, the question is whether or not our dataset is linearly separable after applying ϕ to our original data points. With this in mind, we refer to Thomas Cover's work [4], which shows that non-linearly separable data is very likely to become linearly separable when projected into a higher dimensional space. So long as our feature space is sufficiently larger in terms of dimensions than our input space, we should be able to achieve linear separation.

This approach, however, leads to some problems when considering Problem 4.1 part b). Specifically, for an infinite dimensional $\mathcal{H}$, we may be unable to store the values of $\phi(x)$ in computer memory. Similarly, approximations will need to be made to compute the inner product in the feature space. To circumvent these issues, we introduce kernel functions.

Definition 4.2 (Reproducing Kernel)**.** Let $\mathcal{X}$ be an arbitrary, non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **kernel** if it can be represented as $k(x, z) = \langle \phi(x), \phi(z) \rangle$, where $x, z \in \mathcal{X}$ and $\phi : \mathcal{X} \to \mathcal{H}$.

The function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **reproducing kernel** corresponding to a Hilbert Space $\mathcal{H}$ if the following are satisfied:

1. The function $k(x, \cdot)$, where x is fixed, is an element of $\mathcal{H}$.
2. $f(z) = \langle f, k(z, \cdot) \rangle \;\; \forall z \in \mathcal{X}$

Remark 4.3. Kernel functions are symmetric (conjugate symmetric for complex RKHS) and positive-definite, which is why many texts refer to them as "positive-definite kernels". We do not offer a direct proof of these properties, however, because it should follow immediately from the fact that kernels can be represented as inner products.

Theorem 4.4. *If $\mathcal{H}$ is a RKHS on $\mathcal{X}$, then there exists a unique reproducing kernel, k, that corresponds to $\mathcal{H}$. Conversely, if k is a reproducing kernel, there exists some RKHS, $\mathcal{H}$, corresponding to k.*

Proof. In this paper we will prove the forward direction. We begin by showing that for any RKHS, a reproducing kernel exists.

Let $\mathcal{H}$ be defined as in the theorem statement. Using the reproducing property, there exists a unique $K_x \in \mathcal{H}$ for each fixed $x \in \mathcal{X}$ such that $\mathcal{L}_x(f) = \langle f, K_x \rangle$ for all $f \in \mathcal{H}$. In particular we have

$$\begin{aligned} f(x) &= \mathcal{L}_z(f) = \langle f, K_z \rangle \\ \implies K_x(z) &= \mathcal{L}_z(K_x) = \langle K_x, K_z \rangle . \end{aligned}$$

Define $\phi(x) = K_x$ and define the kernel $k(x, z) = \langle \phi(x), \phi(z) \rangle$. Fix x and note that

$$\begin{aligned} k(x, \cdot) &= \langle \phi(x), \phi(\cdot) \rangle & \qquad\qquad f(z) &= \langle f, K_z \rangle \\ &= K_x(\cdot) \in \mathcal{H} & &= \langle f, k(z, \cdot) \rangle . \end{aligned}$$

Thus, k is a reproducing kernel. To show that k is unique, suppose that there exists another reproducing kernel on $\mathcal{H}$, k'. If we notate $k'(x, \cdot) = K'_x$, then we have the following:

$$\begin{aligned} ||K_x - K'_x||^2 &= \langle K_x - K'_x, K_x - K'_x \rangle \\ &= \langle K_x, K_x \rangle - \langle K'_x, K_x \rangle - \langle K_x, K'_x \rangle + \langle K'_x, K'_x \rangle \\ &= K_x(x) - K'_x(x) - K_x(x) + K'_x(x) \\ &= 0. \end{aligned}$$

The converse of this theorem is known as the "Moore-Aronszajn" theorem. A full proof of this theorem can be found in [1].

$\square$

Now, we are ready to solve Problem 4.1.

Solution 4.5 (Solution to Problem 4.1)**.** Let $\phi : \mathcal{X} \to \mathcal{H}$ be defined as $\phi(x) = K_x$ for all $x \in \mathcal{X}$. We have already discussed how this ϕ satisfies part a). Remember from Theorem 4.4 that $\langle \phi(x), \phi(z) \rangle$ defines a reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ on $\mathcal{H}$. Thus, our modified dual problem (2) becomes

$$\begin{aligned} \max \ & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_{(i)} y_{(j)} \alpha_i \alpha_j k\left(x_{(i)}, x_{(j)}\right) \\ \text{s.t.} \ & \sum_{i=1}^{n} \alpha_i y_{(i)} = 0. \end{aligned} \tag{3}$$

Optimizing Equation (3) is faster and less memory intensive than optimizing Equation (2) since we are working in a lower-dimensional space. At the same time, Equation (3) will yield the optimal w and b such that $h(x) = \langle w, x \rangle + b$ accurately classifies each point in D.

Remark 4.6. Note that Equation (3) lacks any remnants of the RKHS corresponding to the kernel function. Similarly, there is no longer any need to calculate the inner product. This is what makes the kernel trick so widely useful. Instead of defining a Hilbert Space and an inner product, one need only find a binary function that statisfies Definition 4.2.

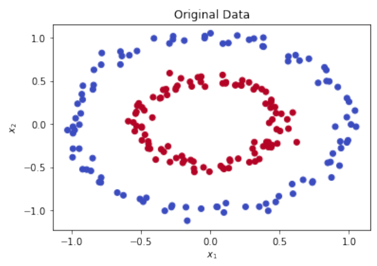
5 Application

In this section, we demonstrate the effect of the kernel trick on non-linearly separable data. We will do so by examining a few types of kernels, visualizing the transformation of data when possible, and comparing the accuracy of base SVM and kernel SVM. For accuracy comparisons, we will use the `SVC` model included in Python's sci-kit learn module. Both the baseline and kernel SVM models will be trained on the same set of 1600 points generated from the distribution of data given within each example. To compute the accuracy, we will test the fitted models on a set of 400 points, again from the same initial distribution. We will start by defining and working with the polynomial kernel.

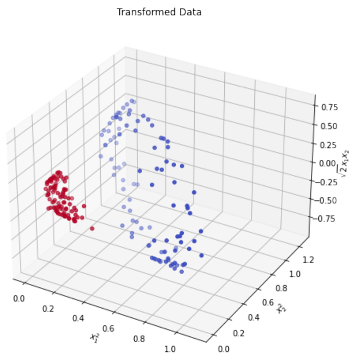
Definition 5.1 (Polynomial Kernel)**.** Let $\mathcal{X} = \mathbb{R}^n$. For some $c \geq 0$ and $d \in \mathbb{N}$, the **polynomial kernel**, k, is defined as $k(x, z) = (\langle x, z \rangle + c)^d$.

We will now use a special case of the polynomial kernel with $d = 2$ and $c = 0$ to demonstrate the kernel trick.

Example 5.2. Consider the dataset given by Figure 1 a.



(a) Non-linearly separable rings



(b) linearly separable rings

Figure 1: Visualizing the feature map ϕ.

Let $\mathcal{X} = \mathbb{R}^2$. Our choice of k is

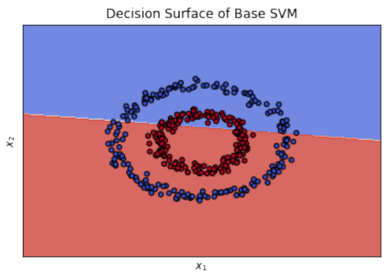
$$\begin{aligned} k(x,z) &= \langle x, z\rangle^2 \\ &= (x_1z_1 + x_2z_2)^2 \\ &= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2. \end{aligned}$$

For this particular kernel, we are able to explicitly find the feature map, ϕ. Note that

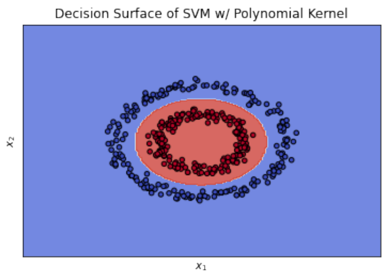
$$x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 = \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1x_2 \end{bmatrix} \begin{bmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{bmatrix}$$

$$\implies \phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}.$$

Now, let us use ϕ to transform our data points. The resulting data points are visualized in Figure 1 b. Clearly, the transformed data is linearly separable in the 3-dimensional feature space. Finally, let us train two SVM models– one using the kernel trick and one not. The achieved results are shown in Figure 2.



(a) Basic SVM misclassifies many points.



(b) The kernel trick allows for 100% accuracy.

Figure 2: Decision boundaries of base/kernel SVM.

In the following example, we examine a type of kernel with an infinite dimensional feature space. Given that we cannot visualize such data, we demonstrate its effectiveness through accuracy comparisons.

Definition 5.3 (Gaussian Kernel)**.** For some $\gamma > 0$, we define the **Gaussian kernel** as $k(x,z) = \mathbf{exp}\left\{\gamma||x-z||^2\right\}$.

	Base SVM	**Kernel SVM**
Accuracy	53%	100%

Table 1: Accuracy with and without kernel trick.

In our example, we will consider the case where $\gamma = 1$.

Example 5.4. Consider the dataset given by Figure 3.

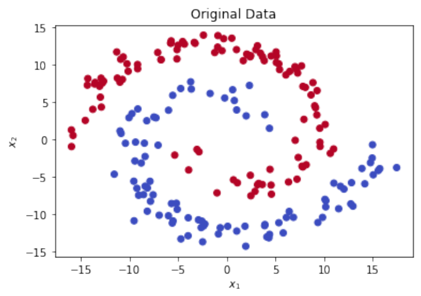
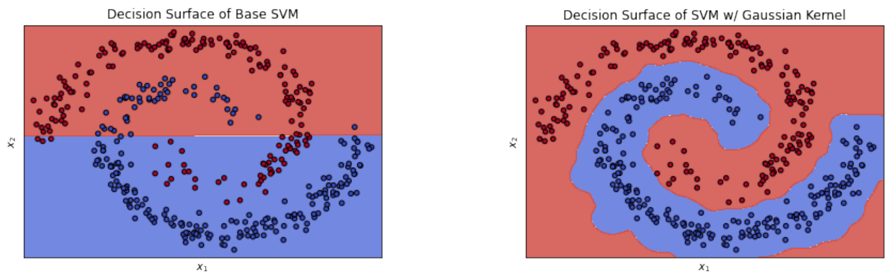


Figure 3: Non-linearly separable spirals.

After training both SVM models, we achieve the following results.



(a) Again, many points are misclassified.

(b) The Gaussian kernel adapts well to complex data.

Figure 4: Decision boundaries of base/kernel SVM.

	Base SVM	**Kernel SVM**
Accuracy	75%	100%

Table 2: Accuracy with and without kernel trick.

References

[1] N. Aronszajn. “Theory of reproducing kernels”. In: 68.3 (Mar. 1950), pp. 337–337. DOI: 10.1090/s0002-9947-1950-0051437-7. URL: https://doi.org/10.1090/s0002-9947-1950-0051437-7.

[2] Philippe Blanchard and Erwin Brüning. “Geometry of Hilbert Spaces”. In: Birkhäuser Boston, 2003, pp. 199–210. DOI: 10.1007/978-1-4612-0049-9_15. URL: https://doi.org/10.1007/978-1-4612-0049-9_15.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge, UK New York: Cambridge University Press, 2004. ISBN: 0521833787.

[4] Thomas M. Cover. “Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition”. In: EC-14.3 (June 1965), pp. 326–334. DOI: 10.1109/pgec.1965.264137. URL: https://doi.org/10.1109/pgec.1965.264137.

[5] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: 2.4 (Dec. 1989), pp. 303–314. DOI: 10.1007/bf02551274. URL: https://doi.org/10.1007/bf02551274.

[6] G. B. Folland. *Real analysis : modern techniques and their applications*. New York: Wiley, 1999. ISBN: 0-471-31716-0.

[7] Alexander G. Ramm. “A simple proof of the closed graph theorem”. In: 4.1 (Dec. 2015), p. 1. DOI: 10.14419/gjma.v4i1.5534. URL: https://doi.org/10.14419/gjma.v4i1.5534.

[8] Karen Saxe. *Beginning functional analysis*. New York: Springer, 2002. ISBN: 0-387-95224-1.

[9] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer New York, 2008. DOI: 10.1007/978-0-387-77242-4. URL: https://doi.org/10.1007/978-0-387-77242-4.